VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

OVLÁDÁNÍ POLOHOVACÍHO ZAŘÍZENÍ PRO AKUSTICKÁ MĚŘENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ PYTELA

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMUNICATIONS

OVLÁDÁNÍ POLOHOVACÍHO ZAŘÍZENÍ PRO AKUSTICKÁ MĚŘENÍ

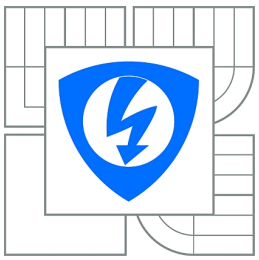
CONTROL OF POSITIONING SYSTEM FOR ACOUSTICAL MEASUREMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — ONDŘEJ PYTELA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SCHIMMEL, Ph.D
SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Ondřej Pytela	***ID:***	133140
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Ovládání polohovacího zařízení pro akustická měření

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte automatické ovládání 2D polohovacího zařízení tvořeného dvojicí lineárních os Berger Lahr PAS41BR pro akustická měření. Zařízení bude ovládáno pomocí rozhraní USB z osobního počítače a bude umožňovat umístění držáku do zadané pozice s přesností odpovídající přesnosti daných lineárních os a lineární posun z pozice A do pozice B zadanou rychlostí. Pro komunikaci s lineárními osami využijte knihovnu ActiveX dodávanou výrobcem. Pro předvedení funkce ovládání vytvořte v jazyce VBA, např. v prostředí MS Excel, jednoduchou aplikaci pro mapování akustického tlaku.

DOPORUČENÁ LITERATURA:

[1] Šnorek, M. Standardní Rozhraní PC 1. vyd. Praha: Grada, 1992. ISBN 8085424800
[2] ILS1B, ILS1F, ILS1R - Lexium Integrated Drive Product manual V2.01. Berger Lahr, 2008
[3] ILx1R RS485 Fieldbus interface manual, v2.01. Berger Lahr, 2008.
[4] Lexium CT Commissioning software manual, v2.04. Berger Lahr, 2009.
[5] Smetana, C. a kol., Hluk a vibrace, měření a hodnocení. Sdělovací technika, Praha 1998. ISBN: 80-901936-2-5

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: Ing. Jiří Schimmel, Ph.D.
Konzultanti bakalářské práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se zabývá řešením problematiky dálkově řízeného polohovacího zařízení, využívajícího krokové motory ILS1R573PB1A0 pohánějící lineární osy Berger Lahr PAS41BR. Krokové motory jsou vybaveny rozhraním RS485 pro komunikaci s ostatními zařízeními a proto byl vyroben příslušný převodník připojitelný přes USB port počítače. Hlavním cílem práce bylo vytvoření řídicí aplikace s použitím knihovny MDC_ActiveX a vývojového prostředí Visual Basic 6.

KLÍČOVÁ SLOVA

ILS1R573PB1A0, RS485, krokový motor, polohovací zařízení, MDC_ActiveX, Visual Basic, mapování akustického tlaku

ABSTRACT

This bachelor's thesis deals with issues of remotely controlled positioning system which uses stepper motors ILS1R573PB1A0 to drive linear axes Beger Lahr PAS41BR. Stepper motors contains RS485 interface for comunication with other devices so appropriate converter was made to make a connection with personal computer via USB port. The main objective was to develop control application by using MDC_ActiveX library and development environment Visual Basic 6.

KEYWORDS

ILS1R573PB1A0, RS485, stepper motor, positioning system, MDC_ActiveX, Visual Basic, acoustical measurements

PYTELA, Ondřej *Ovládání polohovacího zařízení pro akustická měření*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 46 s. Vedoucí práce byl Ing. Jiří Schimmel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Ovládání polohovacího zařízení pro akustická měření“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Jiřímu Shimmelovi, Ph.D. za odborné konzultace a poskytnutí pracovních prostředků a výpočetní techniky k práci.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

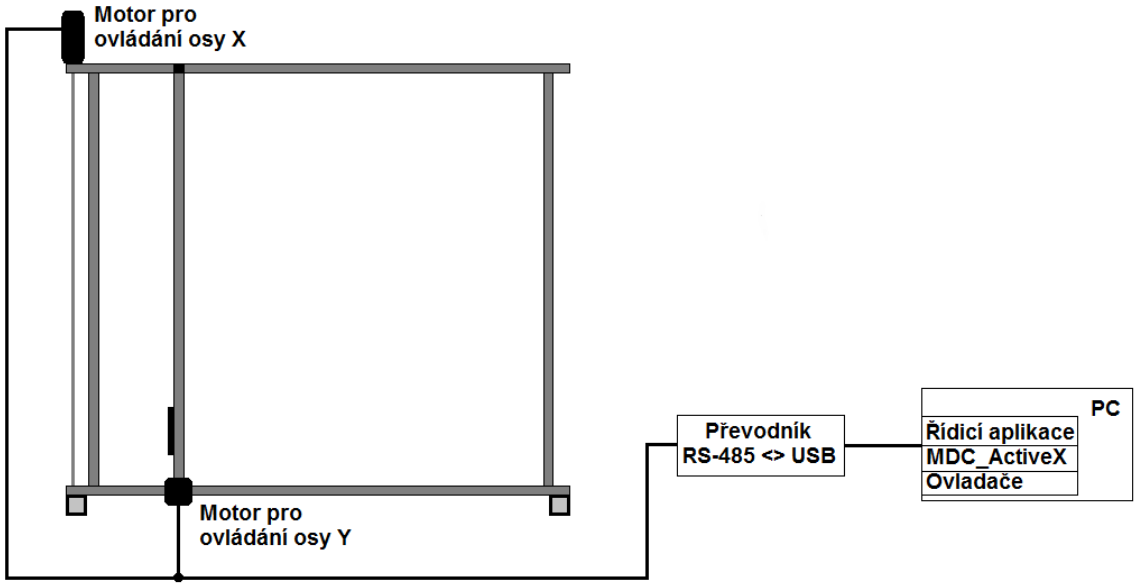
SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této semestrální práce je dálkové ovládání polohovacích os, určených pro akustická měření, prostřednictvím USB portu počítače. Polohovací osy jsou poháněny krokovými motory ILS1R573PB1A0 s integrovanou výkonovou elektronikou a řídícím protokolem využívající standard RS-485 pro komunikaci s ostatními zařízeními.

V práci je zahrnuta výroba USB - RS-485 převodníku, aby bylo vůbec možné komunikaci mezi počítačem a krokovým motorem navázat. Další hlavní částí práce je seznámení se s příkazy a jejich datovou strukturou pro ovládání různých funkcí motoru, zachycení komunikace na USB portu počítače a dekódování zachycených dat. Pro automatické ovládání zařízení byla ve vývojovém prostředí programu Visual Basic 6, s použitím knihovny MDC_ActiveX dodávané výrobcem, vytvořena aplikace pro mapování akustického tlaku.



Obr. 1: Blokové zapojení polohovacího zařízení.

1 MĚŘENÍ AKUSTICKÉHO VÝKONU

K měření akustického výkonu se nějčastěji používá intenzitní sonda. Je zpravidla tvořena dvojicí kvalitních kondenzátorových mikrofonů, které měří hodnoty akustického tlaku ve dvou blízkých bodech. Proto by měly mít použité mikrofony co nejshodnější přenosové funkce. Bývají často uspořádány čely k sobě, což vede ke zlepšení fázové charakteristiky. Pro přesné vymezení jejich vzdálenosti je vložena distanční vložka.

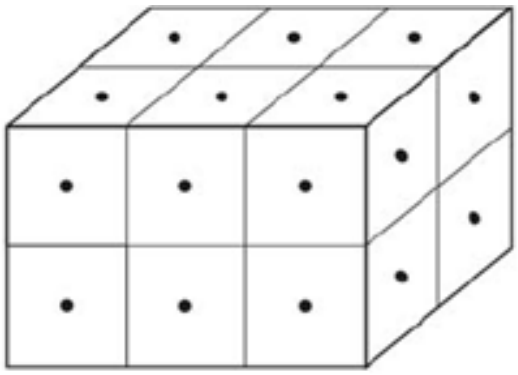
Intenzitní sondou je možné měřit jen v jednom bodě prostoru, takže je třeba použít praktický postup, který umožní změření celého prostoru. Měření akustického výkonu intenzitní sondou lze provést pomocí dvou metod, metodou měření v bodech a metodou měření skenováním.

1.1 Metoda měření v bodech

U této metody je důležité rozdělit si měřené plochy na N částí o ploše S_i a ve středu každé plochy změřit odpovídající intenzitu I_n kolmou k této ploše. Stanovení akustického výkonu je pak prováděno pomocí součtu jednotlivých hodnot intenzit [7]

$$W = \sum_{i=1}^{N} I_{ni} S_i, \tag{1.1}$$

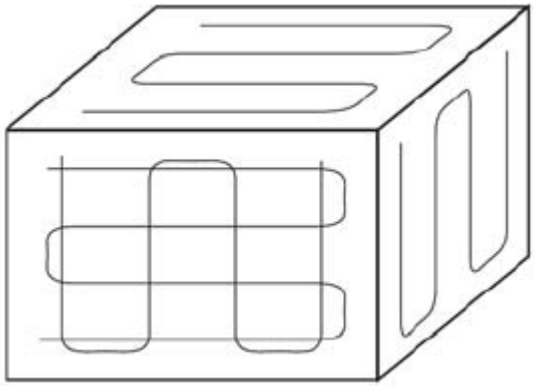
kde I_{ni} je normálová intenzita měřená intenzitní sondou uprostřed elementu plochy S_i a N je počet elementů, na které je měřicí obalová plocha rozdělena.



Obr. 1.1: Metoda měření v bodech [7].

1.2 Metoda měření skenováním

Pokud se sonda bude pohybovat spojitě po dráze na měřící ploše, jedná se o metodu skenování. Rychlost pohybu sondy by měla být menší než 1 m/s a měla by být konstantní [7].



Obr. 1.2: Metoda měření skenováním [7].

Samotné provedení měření obou metod je usnadněno upevněním intenzitní sondy k držáku polohovacího zařízení, které automaticky přemisťuje intenzitní sondu do zadaných pozic.

2 PŘIPOJENÍ ZAŘÍZENÍ

2.1 Rozhraní RS-485

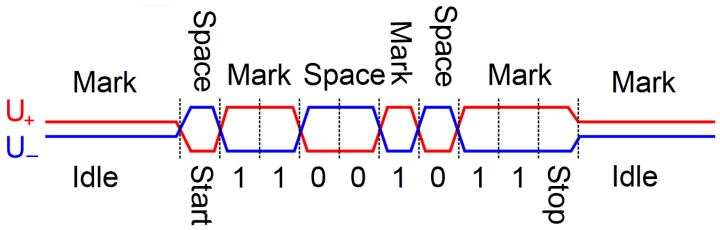
Standard sériové komunikace RS-485 má stejný základ jako široce rozšířený standard RS-232, od kterého se liší především jinou definicí napěťových úrovní. Je navržen tak, aby umožňoval vytvoření poloduplexního dvouvodičového vícebodového sériového spoje, ke kterému je možné připojit až 32 zařízení s možností komunikace na vzdálenost až 1200 m. Vodiče se označují jako „A“ a „B“. V klidovém stavu přenosové linky by mělo být na vodiči A menší napětí než na vodiči B, proto se používá i označení vodičů „-“ a „+“. Přenosová rychlost kanálu může být na krátké vzdálenosti (jednotky metrů) až 10 Mb/s a je nepřímo úměrná délce vedení . Při komunikaci na větší vzdálenosti musí být vedení zakončeno tzv. *terminály*. Jedná se o zakončovací rezistory (ideálně velikosti 110 Ω) na koncích vedení tak, aby výsledná impedance linky byla 55 Ω. Tím se zvýší odolnosti linky proti rušení a zabrání se odrazům signálu od konců vedení.

2.1.1 Logické úrovně

Změna logické úrovně je prezentována změnou napětí mezi vodiči A a B, což má za následek dobrou odolnost proti indukovanému rušení do vedení. Logický stav „1“ (také označován jako *Mark*) je indikován rozdílným napětím na vodičích A - B < -200 mV, naopak logický stav „0“ (*Space*) odpovídá napětí A - B > +200 mV.

2.1.2 Přenos dat

Samotný přenos dat je realizován sedmi nebo osmi bitovými rámci doplněných vždy o star bit, který je reprezentován logickou nulou. Na konci rámce je vyslán alespoň jeden stop bit a případně je doplněn ještě paritní bit.



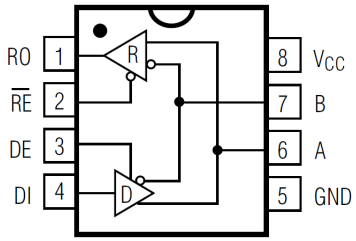
Obr. 2.1: Ukázka přenosu čísla 11010011b se start a stop bitem.

3 KONSTRUKCE PŘEVODNÍKU

Pro ovládání lineárních os bylo zapotřebí vyrobit převodník, který bude řešit přenos komunikace mezi USB rozhraním osobního počítače a samotným motorem s protokolem RS-485. Řešením bylo použití integrovaného obvodu od firmy FTDI, FT232RL [4]. Jedná se o převodník z USB rozhraní na sériový UART port, který má hned několik výhod. Hlavními jsou:

- Implementovaný USB protokol, není třeba zvlášť programovat firmware
- Konfigurovatelné výstupní I/O piny
- Integrovaná EEPROM paměť pro uložení popisu zařízení a uložení konfigurace výstupních pinů
- Podpora UART přenosu se sedmi nebo osmi bitovými rámci, s více stop bity a paritami
- Příjem/vysílání dat může být indikován připojenými LED
- Modulační rychlost přenosu dat až 3 Mbd

Výstupem z převodníku je klasický RS-232 standard. Jak již bylo zmíněno předešlé kapitole, úpravou napěťových úrovní lze vytvořit požadovaný RS-485 standard. Následuje tedy druhý převodník, MAX481CSA [3]. Ten obsahuje de facto dvě oddělené části. Jednu pro příjem a druhou pro vysílání dat. Činnost těchto částí je závislá na logickém stavu na řídících pinech 2 (DE) a 3 ($\overline{\text{RE}}$). Zapojením tohoto obvodu vznikne celkový převodník pro komunikaci mezi počítačem a motorem.



Obr. 3.1: Struktura převodníku MAX481CSA [3].

3.1 Schéma zapojení

Schéma zařízení (viz příloha A.1) vychází z katalogového zapojení výše zmíněných integrovaných obvodů [4]. USB vstup do převodníku je tvořen prostřednictvím USB-B konektoru, ze kterého je celý převodník také napájen. Napájecí napětí je vyhlazeno a odrušeno kondenzátory C4 a C5. K IC1 jsou přes rezistory limitující proud zapojeny LED symbolizující vysílání (žlutá) a příjem dat(červená) vzhledem k motoru. Pokud bychom přijímací část obvodu IC2 (pin 2 (DE)) vypnuli přivedením log.

0 a zároveň bychom na pin 3 ($\overline{\text{RE}}$) přivedli log. 1 (vypnutí vysílací části), obvod MAX481CSA se přepne do režimu velmi nízké spotřeby. Tento režim ale nebude využíván, proto jsou piny 2 (DE) a 3 ($\overline{\text{RE}}$) spojeny a jsou zároveň řízeny signálem z CBUS2 obvodu IC1, který je standardně nastaven tak, aby při vysoké logické úrovni byla aktivní vysílací část. Rezistory R4 a R5 napomáhají k lepší stabilitě napěťových úrovní, R6 a R7 chrání linku proti přetížení. Standard RS-485 nemá specifikované zapojení konektorů a proto jsou řídící signály A a B společně se zemí vyvedeny na šroubovací svorkovnici. Mezi řídící signály lze pomocí jumperu zapojit terminálový rezistor R8.

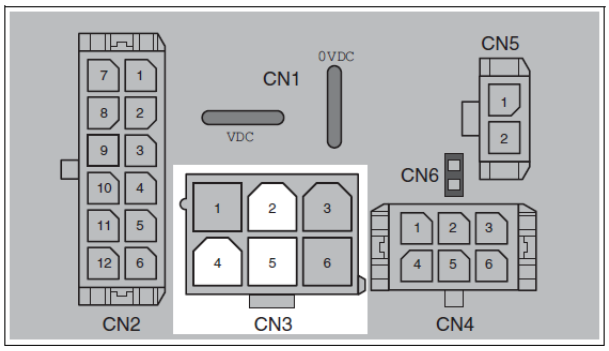
3.2 Sestavení převodníku

Plošný spoj je převážně osazen součástkami SMD, seznam součástek je uveden v tabulce A.1. Pro snadné připojení převodníku jsou konektory umístěny na krajích krabičky, opatřené otvory pro jejich vyvedení. Na horní straně krabičky jsou pak vyvrtány díry pro vyvedení signalizačních LED a díry pro přístup šroubováku ke svorkovnici na straně RS-485 protokolu bez nutnosti demontáže krytu krabičky.

4 KROKOVÝ MOTOR

4.1 Rozhraní motoru

Krokový motor ILS1R573PB1A0 obsahuje různé konektory, které jsou znázorněny na obr. 4.1. Napájecí napětí pro motor je stejnosměrné, v rozsahu 24 - 40 V a je přivedeno přes konektory CN1 (+) a CN2 (gnd). Proudový odběr motoru může dosáhnout špičkově až 3,5A. Kabeláž umožňující komunikaci protokolem RS-485 je připojena prostřednictvím konektoru CN3, kde na pinu 2 je signál B (+), na pinu 5 je signál A (-) a na pin 4 je vyvedena zem pro stínění komunikační kabeláže [2]. Konektor CN4 slouží pro připojení doplňkových zařízení či indikátorů, jako jsou například dorazová čidla a čidla referenčních bodů.

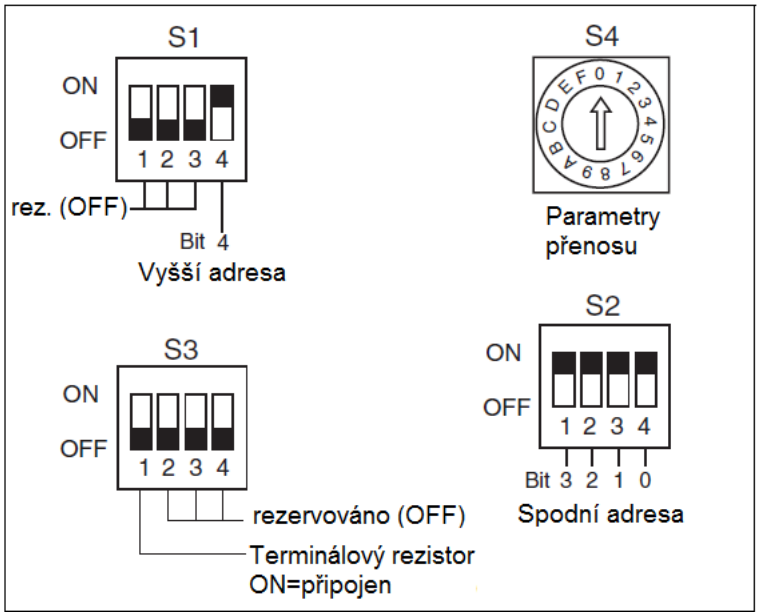


Obr. 4.1: Nákres konektorů [1].

4.2 Navázání komunikace

Typická topologie sériových linek využívajících protokol RS-485 obsahuje jedno řídící zařízení, tzv. *Master* (Osobní počítač, PLC modul,..), které zároveň s obslužným softwarem řídí dění na lince, ke které jsou připojeny zařízení naslouchající, tzv. *Slave*. Takových jednotek může být připojeno k sériové lince až 32 a proto musí mít každé toto zařízení přidělenou adresu. Krokovému motoru lze přidělit adresu pomocí konfigurace DIP přepínačů (obr. 4.2) umístěných v prostoru pod konektory, zbylými DIP přepínači lze volit zařazení terminálového rezistoru v případě, že je zařízení umístěno na konci linky [2]. V prostoru pod konektory je i umístěn otočný přepínač určující parametry přenosu, viz tab. 4.1.

Motoru byla přiřazena adresa 1 nastavením DIP přepínačů do polohy OFF kromě přepínače symbolizujícího nultý bit. Terminálový rezistor je příslušným přepínačem v poloze OFF odpojen a parametry přenosu byly zvoleny otočným přepínačem do pozice 0 (9600 Bd, 7 bytových rámců, sudá parita, jeden stop bit).



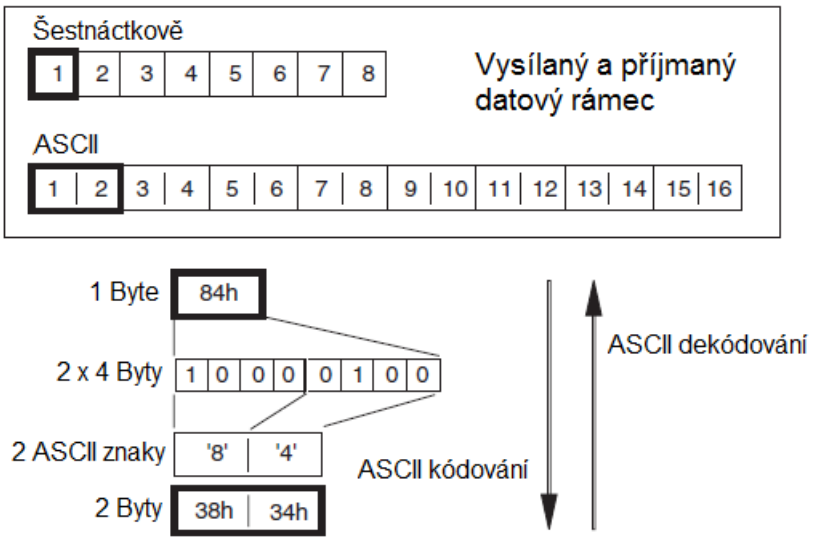
Obr. 4.2: Přepínače pro nastavení adresy a parametrů přenosu [1].

Tab. 4.1: Volba parametrů přenosu [1]

Poloha přepínače	**Modulační rychlost**	**Parametry**
0	9600	7-E-1
1	19200	7-E-1
2	38400	7-E-1
3	-	-
4	9600	7-N-1
5	19200	7-N-1
6	38400	7-N-1
7	-	-
8	9600	8-E-1
9	19200	8-E-1
A	38400	8-E-1
B	-	-
C	9600	8-N-1
D	19200	8-N-1
E	38400	8-N-1
F	-	-

4.2.1 Tvar příkazů

Krokové motory a obslužný program v počítači používají na vysílání dat ASCII formát zastoupený 16-ti poli. V případě vysílání dat v hexadecimálním tvaru je nutné hexadecimální tvar zastoupený 8-mi poli převést do formátu ASCII. Princip kódování je popsán v následujícím obrázku 4.3.



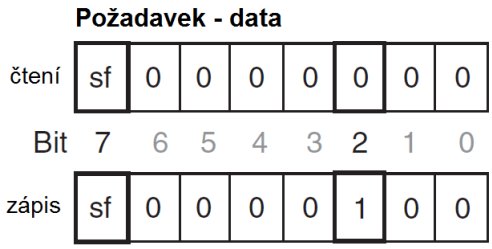
Obr. 4.3: Princip ASCII kódování [2].

Číslo 84h obsahující 8 bitů se rozdělí na poloviny, kde vzniklé 4-bitové části tvoří základ pro určení dvou ASCII znaků , v tomto případě 8 a 4. Tyto znaky se podle pozice v ASCII tabulce zakódují do šestnáctkového tvaru, čímž z původního šestnáctkového čísla 84h vzniknou čísla dvě, 38h a 34h. Za 16ti bytový ASCII rámec se doplní číslo 0Dh (<CR> Carriage return, často reprezentováno stiskem klávesy Enter) jako symbol ukončení rámce. Adresováním motoru příkazem #01 <CR>, v hexadecimálním tvaru jako 23h, 30h, 31h, motor odpoví stejnou zprávou (pokud souhlasí nastavení adresy) a je navázána tzv. *point-to-point* komunikace, při které můžeme vysílat další příkazy. Vzniklý komunikační kanál můžeme zrušit adresováním jiného zařízení. Struktura dat je popsána v následující tabulce 4.2.

První byte nesoucí kontrolní informace (**požadavek**) je složen z 8mi bitů (obr. 4.4), pomocí kterých zařízení master zajišťuje synchronizaci spojení, zasílá zprávu pro potvrzení přijatých dat a uvádí, o která data se jedná. Druhy těchto dat jsou dva. Zařízení master může od motoru zjišťovat data, nebo data zadávat. Příkladem zjišťování dat je například čtení aktuální pozice motoru, jeho teploty, napájecího napětí apod. Zadávanými daty jsou příkazy k vykonání pohybu, zapnutí či vypnutí výkonové elektroniky nebo nastavení parametrů pohybu.

Tab. 4.2: Datová struktura příkazů [2].

Pořadí bytu		**Popis**
ASCII	**HEX**	
1-2	1	Kontrolní informace následujícího příkazu
3-4	2	Subindex příkazu
5-8	3-4	Index příkazu
9-16	5-8	Data příkazu



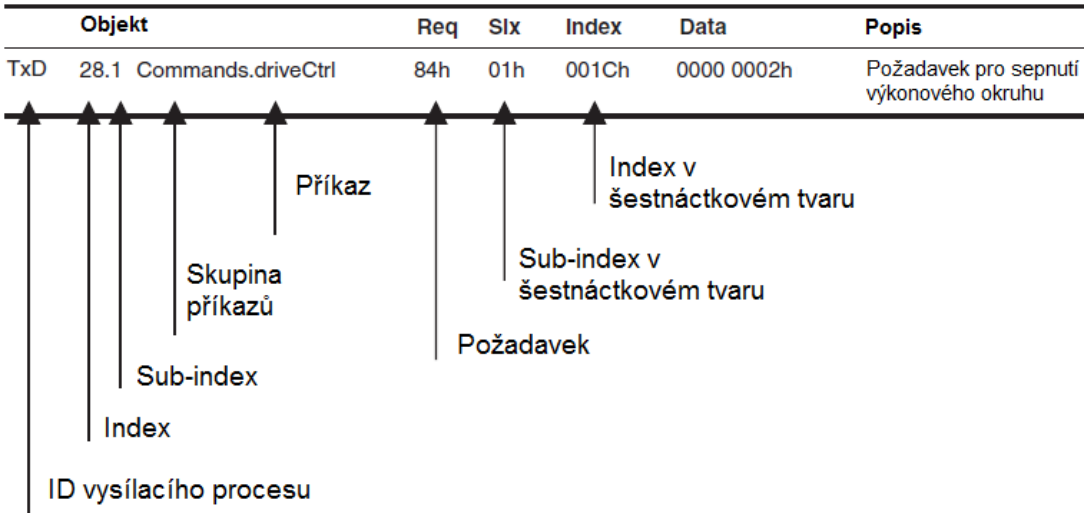
Obr. 4.4: Struktura bytu s požadavkem [2].

Pokud jde o nový typ příkazu, master změní bit číslo 7 (**sf**) na hodnotu 1. Bitem číslo 2 master určuje, pokud jde o data pro zápis, nebo chce data z motoru číst. Na hodnotách ostatních bitů nezáleží a proto může požadavek nabývat v podstatě libovolných hodnot.

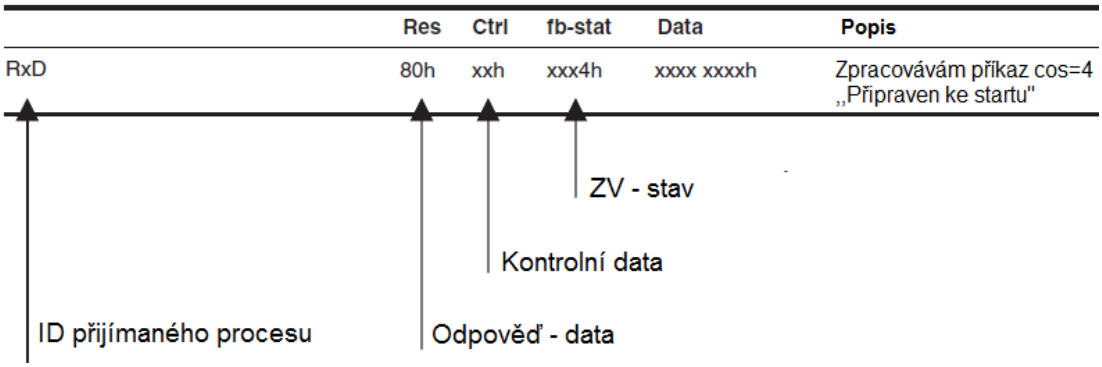
Každý z příkazů má pro přehlednost svůj specifický název v textové podobě, který má přidělen svůj číselný **Index** a subindex (**Six**), pomocí kterého jsou příkazy předávány motoru. Tím motor rozpozná, o jaký příkaz se jedná. Pomocí samotných dat jsou pak příkazy doplněny o požadované nastavitelné hodnoty. Příklad kompletního příkazu je uveden na obrázku č. 4.5.

Příkaz v poli **Object** má index a subindex uveden jako **28.1** v dekadickém tvaru. Číslo 28 je po přepočtu do tvaru hexadecimálního 1Ch, protože pole pro index musí být zastoupeno čtyřmi pozicemi, musí být index v tomto případě **001Ch**. Obdobně se postupuje pro určení sub-indexu, který má ovšem vymezeny pozice pouze dvě.

Při zpětné odpovědi motoru směrem k počítači je na prvním místě datové struktury byte s názvem odpověď-data (**Res**), který stavem svých bitů řídí synchronizaci, potvrzení příkazů a oznámení o výskytu chyb. Sub-index je nahrazen kontrolními daty (**Ctrl**) a index je nahrazen hodnotou s názvem **fb-stat**, která činí zpětnou vazbu a vypovídá o aktuálním stavu motoru. Tvar struktury dat vysílané motorem je znázorněn na obrázku č. 4.6.



Obr. 4.5: Struktura vysílaného příkazu [2].



Obr. 4.6: Struktura přijímaného příkazu [2].

4.3 Zápis příkazů

Pro zápis příkazů do motoru je třeba nastavit odpovídající bity v datové struktuře **požadavek**. Pokud budeme uvažovat nulové hodnoty bitů a budeme nastavovat jen příslušné bity dle předešlé kapitoly, pro nový příkaz zapsaný do motoru může být hodnota **požadavku** = 84h.

4.3.1 Operační stav motoru

Parametrem **28:1 Commands.driveCtrl** [1] se mění aktuální stav motoru, který může být volen dle tabulky č. 4.3. Stav, ve kterém se může motor nacházet je zadán posledními 4mi bity v poli pro **Data**. Pokud bychom chtěli zapnout výkonovou část motoru, správný tvar příkazu bude XX01001C00000002h. XX zastupuje hodnotu pro požadavek.

Tab. 4.3: Operační stavy motoru [2].

Pořadí bitu				HEX	Kontrolní slovo	Stav
3	**2**	**1**	**0**			
0	0	0	1	0001h	Disable	Vypnutí výkonové části
0	0	1	0	0002h	Enable	Zapnutí výkonové části
0	1	0	0	0004h	Quick stop	Rychlé zablokování
1	0	0	0	0008h	Fault Reset	Resetování motoru

4.3.2 Referenční rychlost

Při pohybu lineárních os zadanou rychlostí musíme motoru určit počet otáček za minutu. Krokový motor má velmi jemné rozlišení v počtu 20000 kroků na jednu otáčku, motor je schopen dosáhnout maximální rychlosti otáčení až 3000 ot/min. Převod motoru vzhledem k pozici osy odpovídá 2500 krokům pro posun o 10 mm. Pokud budeme chtít vykonat pohyb osy rychlostí např. 320 mm/min, motor musí za minutu vykonat pootočení hřídele o 80000 kroků. Tato hodnota se přepočte na počet otáček za minutu (80000/20000 = 4) a zadává se opět v přepočtu do hexadecimálního tvaru (4 = 4h) pomocí příkazu **35:5 PTP.v_tarPTP**[1]. Sub-index příkazu je potom roven hodnotě 05h, index 0023h a data budou mít hodnotu 0000 0004h, výsledná podoba příkazu je XX05002300000004h. Motor disponuje rozlišením 1 ot/min.

4.3.3 Akcelerace, decelerace

Krokový motor má úctyhodný kroutící moment a jeho prudký pohyb může vyvolat hned několik nepříjemností. Při posouvání těžšími předměty může snadno dojít důsledkem skokové změny rychlosti k proudovému přetížení motoru. V neposlední řadě při nastavení vysoké referenční rychlosti může v konstrukci polohovacích os s připevněným mikrofonem docházet k cukání a doznívajícímu vibrování, což se projeví na výsledné kvalitě měření. V extrémním případě hrozí poškození mechanických dílů celé konstrukce.

Řešením této problematiky je nadefinování akcelerace (decelerace) při provádění pohybu pomocí příkazu **29:26 Motion.acc** [1]. Příkaz vychází z nulové a referenční hodnoty rychlosti. Data příkazu vyjadřují, o kolik kroků (Inc) za minutu se navýší aktuální rychlost otáčení, než dospěje k referenční rychlosti (v případě decelerace jde o postupný pokles kroků za minutu od referenční rychlosti k rychlosti nulové). Rozsah nastavitelné akcelerace/decelerace je v rozmezí 0 - 765000 kroků za minutu.

Převedením požadované hodnoty zrychlení, např. přídavek 500 kroků za minutu, do šestnáctkového tvaru (1F4h), zjistíme hodnotu zadávaných dat. Parametr zadávající zrychlení či zpomalení pak dostane podobu XX1A001D000001FAh.

4.3.4 Směr otáčení

Motor připočítává zadaný počet kroků (Inc) od referenčního bodu ve výchozím směru otáčení.

Výchozí směr otáčení motoru by se měl volit pouze jednou před samotným vykonáváním pohybů. Volí se parametrem **28:6 Motion.invertDir** [1], který v poli pro **Data** může nabývat pouze dvou hodnot, podle směru otáčení. Poslední byte o hodnotě 1 znamená otáčení proti směru hodinových ručiček (při pohledu na motor směrem od zadní části s konektory k hřídeli), hodnota 0 značí otáčení po směru hodinových ručiček a je to také výchozí nastavení po spuštění motoru. Subindex příkazu **28:6 Motion.invertDir** má hodnotu 06h, index 1Ch a pole pro Data bude v hodnotě 0. Výsledná podoba příkazu pro výchozí pohyb ve směru hodinových ručiček je XX06001C00000000h.

Během pohybu motoru se aktuální směr otáčení mění dle hodnoty zadané pozice. Pozice může být zadána v absolutní míře (je vztažena k referenčnímu bodu) nebo v relativní míře, kdy motor přičítá zadaný počet kroků k aktuální pozici.

4.3.5 Absolutní polohování

Při sepnutí motoru jsou hodnoty aktuální pozice vynulovány a jsou brány jako referenční. Výchozí pozici můžeme snadno změnit posunutím motoru do jakékoliv polohy (např. krajní polohy indikované dorazovým čidlem) a zadáním parametru **28:1 Commands.driveCtrl** v příslušném tvaru provést resetování motoru.

Absolutním polohováním můžeme krokovým motorem přesunout polohovací osu na zadanou pozici vyčíslenou v počtu kroků (Inc). Toto polohování zajišťuje parametr **35:1 PTP.p_absPTP** [1], který v poli pro **Data** bude obsahovat šestnáctkově vyjádřenou požadovanou pozici. Budeme-li chtít od výchozí pozice posunout polohovací osu na vzdálenost 500 mm, musíme přepočtem určit výsledný počet kroků. Jak již bylo zmíněno, pro posun o 10mm je třeba 2500 kroků, v případě vzdálenosti 500mm to bude (125000 = 1E848h) kroků. Subindex a index příkazu opět určíme vyjádřením čísla **35:1** v hexadecimálním tvaru. Tvar příkazu pro přesun na požadovanou pozici bude XX0100230001E848h. Motor automaticky mění směr otáčení dle zadané polohy.

4.3.6 Relativní polohování

Relativní polohování vyjadřuje přídavek kroků k aktuální (absolutní) pozici. Díky svému charakteru bude tento typ polohování hrát největší roli při požadovaných akustických měřeních, kdy se zadá absolutní poloha motoru a následně pravidelně se opakující relativní přídavky kroků pro jednotlivé body měření.

Relativní polohování se tvoří pomocí příkazu **35:3 PTP.p_relPTP** [1]. Podobně jako u absolutního polohování, pole pro **Data** bude obsahovat údaje o množství kroků, které se mají k aktuální pozici připočítat. Při pohybu osy např. o 20 mm bude třeba vykonat motorem, ve výchozím směru otáčení, 5000 kroků. Vyjádřeno šestnáctkově, počet kroků bude roven 1388h. Subindex parametru pro relativní polohování je 03h, index 0023h a data 00001388h. Pro objasnění je opět uveden výsledný tvar příkazu, XX03002300001388h.

4.4 Čtení parametrů

Na každý zadaný parametr zasílá motor odpověď o provedení příkazu, nebo případné chybové hlášení.

Při zjišťování parametrů jsou použity příkazy se stejnou datovou strukturou, jako při zadávání příkazů. Protože se jedná o čtení dat a nikoliv zápis, 2. bit pole **požadavek** bude vynulován a za předpokladu pouze 7. bitu v log.1 bude první dvojice odesílané datové struktury rovna 80h. Každý příkaz má opět svůj index a sub-index, pole s daty je v tomto případě nulové. Na vyslané příkazy motor odpovídá strukturou dat odpovídající obr. 4.6. Sedmý bit bytu **odpověď-data** mění svůj logický stav tak, aby odpovídal přijatému log. stavu vyslaném sedmým bitem bytu **požadavek** při přijímání příkazu. Dokud nenastane shoda těchto bitů, zadaný příkaz nebyl motorem proveden. Šestý bit bytu **odpověď-data** symbolizuje hlášení chyb log. stavem 1, v bezchybném provozu je jeho hodnota nulová.

4.4.1 Monitorování chyb

Na operace zadávané řídícím počítačem motor vždy odpovídá informacemi o svém aktuálním stavu. Během přenosu může dojít ke dvěma druhům chybových hlášení:

- Synchronní chybové hlášení
- Asynchronní chybové hlášení

Synchronní chybové hlášení podává motor ihned po přijmutí příkazu, který z různých příčin nemůže vykonat. Chybová hlášení ukládá motor do registru **ZV - stav** (zadáním příkazu **28:2 Status.driveStat** [1] se vypíše), kde hodnoty příslušných

bitů datové struktury určují, o jaký druh chyby se jedná. Možná chybová hlášení jsou uvedena v tabulce 4.4 a by měla být pravidelně monitorována řídícím programem.

Asynchronní chybové hlášení je motorem odesláno během výskytu závažné chyby, motor se ihned zastaví.

Tab. 4.4: Obsah bytů 3 a 4 (ZV - stav) datové struktury [2].

Bit	**Název**	**Popis**	**Výpis příkazem**
0-3	cos	Operační stav, viz tab. 4.5	
5	FltSig	Interní monitorovací sig.	28:18 Status.FltSig_SR
6	Sign_SR	Externí monitorovací sig.	28:15 Status.Sign_SR
7	warning	Varování	28.10 Status.WarnSig
13	x_add_info	Info referenční pozice	28:3 Status.xMode_act
14	x_end	Ukončení pohybu	28:3 Status.xMode_act
15	x_err	Chyba pohybu	28:3 Status.xMode_act

Tab. 4.5: Výpis operačních stavů motoru [2].

Pořadí bitu				**HEX**	**Popis**
3	**2**	**1**	**0**		
0	0	0	1	0001h	Start
0	0	1	0	0002h	Výkonová část není připravena
0	0	1	1	0003h	Výkonová část vypnuta
0	1	0	0	0004h	Výkonová část připravena
0	1	1	0	0005h	Provádění příslušné operace
0	1	1	1	0006h	Sepnutí rychlobrzdy
1	0	0	0	0007h	Nevyřešení chyb

4.4.2 Aktuální rychlost, pozice motoru

Během vlastního pohybu motoru můžeme příkazem **31:9 Status.n_act** [1] monitorovat momentální rychlost. Odeslání dat v příslušném tvaru je XX06001F00000000. Zjišťovaná hodnota rychlosti při odpovědi motoru je prezentována hodnotou v poli pro data. Pro přehlednost je tato akce zobrazena na obr. 4.7, kde motor odpověděl bez chybových hlášení a v poli pro data vypsal aktuální rychlost v hexadecimálním tvaru.

	Objekt	Req	Six	Index	Data	Popis
TxD	31:9 Status.n_act	80h	06_h	$001F_h$	$0000\ 0000_h$	Čtení aktuální rychlosti motoru

	Objekt	Res	Ctrl	fb-status-word	Data	Popis
RxD		80h	xx_h	$xxxx_h$	0000 03E8h	Aktuální rychlost je 3E8h = 1000 ot/min

Obr. 4.7: Zjišťování aktuální rychlosti motoru [2].

Stejným postupem lez zjistit pozici, na které se motor nachází. Během pohybu motor svou pozici zadává automaticky odpovědí na přicházející zadání požadované pozice. Pokud ale motor stojí, příkazem **31:6 Status.p_act** [1] s výslednou zadávanou datovou strukturou XX06001F00000000h zjistíme jeho pozici. Odpovědí bude hodnota v poli dat vyjádřena šestnáctkově v počtu kroků od referenční pozice.

4.4.3 Provozní hodnoty

Během provozu motoru lze i monitorovat jeho napájecí napětí (příkaz **31:20 Status.UDC_act** [1]), odebíraný proud (příkaz **31:12 Status.I_act** [1]) a teplotu. Motor během své práce může pozastavit činnost vlivem přehřání, zejména pokud pohybuje těžšími předměty proměnlivou rychlostí. Chlazení motoru přispívá nejen rám, do kterého je motor zasazen, ale také vhodný způsob zacházení. Pokud motor nevykonává žádný pohyb, je vhodné (i vzhledem ke spotřebě) vypnout jeho výkonový okruh. Monitorování teploty výkonového okruhu probíhá příkazem **31:25 Status.TPA_act** [1]. U zjišťování napětí či proudu je v odpovědi motoru v poli data zahrnut údaj v šestnáctkové podobě, kde při monitorování je výsledné číslo vyjádřeno v desetinách. Například odpověď na zjišťované napětí má hodnotu 162h, dekadický tvar pak 354, napětí na motoru je potom 35,4V. V případě monitorování teploty je výsledek po převedení přímo udán ve stupních Celsia.

5 REALIZACE KOMUNIKACE

Pro prvotní navázání komunikace byl použit program Lexium CT firmy Schneider Electric. Program podporuje komunikaci velkého množství různých zařízení, včetně použití jiných standardů než právě námi zvolený RS-485. Lexium CT pomocí grafického uživatelského rozhraní interpretuje obsluhou zadávané parametry motoru a zjišťuje řídící data a případné chyby.

5.1 Zachycení komunikace

Zachytávání komunikace bylo provedeno programem Advanced Serial Port Monitor v.3.4 v režimu „Spy“, kterým bylo odposloucháváno dění na příslušném portu počítače od doby, kdy byla programem Lexium CT navázána příkazem adresování komunikace. Následně byl proveden sled příkazů zajišťující pohyb motoru. Program Lexium CT ihned po adresování motoru provedl řadu příkazů dotazujících se na verzi firmwaru motoru, jeho výrobní číslo apod. Pravidelně také četl z motoru chybová hlášení a informace o provozních hodnotách, proto je uvedený výčet (viz příloha B) dění komunikace redukován, aby byla posloupnost příkazů zřetelnější. Vysílání dat značí řádky zakončené písmeny TX, příjem dat potom RX. Před těmito znaky je vepsáno ID zprávy, které vytvořil zachytávací program a nehraje pro nás žádnou významnější roli.

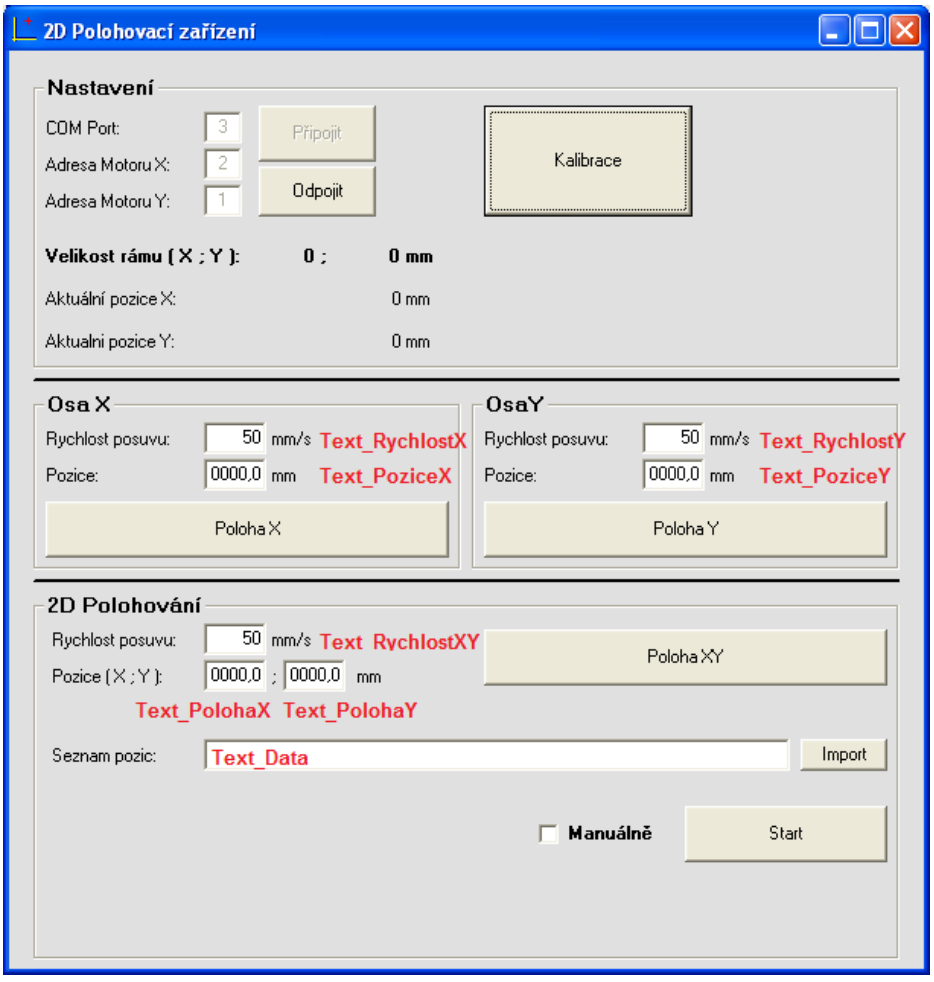
5.2 Vytvoření textového souboru

Na základě odhalení datové struktury příkazů a následném zachycení komunikace byl vytvořen textový soubor s posloupností vhodných příkazů tak, aby bylo možné vykonat jednorázový pohyb motoru. Výchozí parametry pohybu jsou v textovém souboru nastaveny na adresu motoru 1, referenční rychlost 100 ot/min, akcelerace/decelerace 598 kroků/min, výchozí směr otáčení ve směru hodinových ručiček a po sepnutí výkonového okruhu posun do absolutní pozice 200000 kroků. Parametry pohybu lze měnit modifikací dat v souboru, který obsahuje kroky:

1. Navázání komunikace příkazem #XY (XY je nastavená adresa motoru, zde 01),
2. Nastavení referenční rychlosti příkazem **35:5 PTP.v_tarPTP**,
3. Nastavení akcelerace/decelerace příkazem **29:26 Motion.acc**,
4. Nastavení výchozího směru otáčení příkazem **28:6 Motion.invertDir**,
5. Připojení výkonového okruhu příkazem **28:1 Commands.driveCtrl**,
6. Zadání absolutní pozice a start pohybu příkazem **35:1 PTP.p_absPTP**.

6 ŘÍDICÍ APLIKACE

Pro vytvoření aplikace, která bude uživatelským rozhraním usnadňovat ovládání krokových motorů, bylo použito vývojové prostředí programu Visual Basic 6. V tomto prostředí je možné přidávat různé komponenty usnadňující samotné programování. Jednou z použitých komponent je „MDC_ActiveX“, která je poskytována výrobcem motorů a dá se tak s výhodou použít. Obsahuje několik základních parametrů reprezentovaných uživatelsky jednodužšími výrazy, pro některé funkce motoru je nutné použití indexu a sub-indexu příkazu. Voláním instance „MDC1“ lze zadávat pokyny ve tvaru *MDC1.skupina příkazů.konkrétní příkaz číslo zařízení, zápis/čtení, hodnota*. Například nastavení referenční rychlosti posuvu při polohování na hodnotu 60 ot/min nastavíme příkazem ve tvaru **MDC1.OperatingModes.ProfilePosRef ValueSpeed 1, Write_Parameter, 60**. Pro volání funkcí pomocí indexu a sub-indexu je příkaz ve tvaru *MDC1.Config.CallByIndex číslo zařízení,zápis/čtení, index, sub-index, hodnota*. Číslo zařízení je vždy 1.



Obr. 6.1: Okno aplikace s vyznačenými názvy prvků.

Tab. 6.1: Použité příkazy v aplikaci pro řízení motorů [5].

Název	**Účel**	**Poznámka**
Skupina „Config“:		
SetDevice 1, 4, 4, Adresa, Port	Připojení COM portu	
ComPort_Close	Odpojení COM portu	
CallByIndex 1, Write_Parameter, 28, 13, A	Monitorování čidel	A=0 vypnuto A=1 pravé/dolní A=2 levé/horní
CallByIndex 1, Write_Parameter, 40, 3, A	Přidělení nové pozice	[kroky]
CallByIndex 1, Read_Parameter, 31, 20	Čtení napětí	[V] * 0,1
Skupina „Setting“:		
InitDevice 1	Inicializace motoru	
FaultReset 1	Vymazání chyb	
EnablePowerAmplifier 1	Zapnutí výk. okruhu	
DisablePowerAmplifier 1	Vypnutí výk. okruhu	
Skupina „OperatingModes“:		
ProfileVelocity 1, Write_Parameter, A	Neustálé otáčení	[ot/min]
ProfilePosRefValueSpeed 1, Write_Parameter, A	Nastavení referenční rychlosti	[ot/min]
ProfilePosAbsolute 1, Write_Parameter, A	Nastavení pozice	[kroky]
Skupina „ReadValues“:		
Position_ActValue(1) = A	Čtení pozice	[kroky]
Speed_ActValue(1) = A	Čtení rychlosti	[ot/min]

6.1 Připojení

Aplikace umožňuje pro připojení motorů zadat číslo COM portu, adresu motoru pro posuv v ose X („Motor X“) a adresu motoru pro posuv v ose Y („Motor Y“). MDC_ActiveX umožňuje připojení více druhů krokových motorů. Pro řadu ILS1 využívající protokol RS-485 platí druh připojení „Icla“, který je symbolizován číslem 4 a protokol „BLProtokolViaCom“ taktéž s číslem 4 [5]. Po kliknutí na tlačítko

„Připojit“ se program použitím parametru **MDC1.Config.SetDevice 1, 4, 4, Adresa motoru, číslo COM portu** připojí k zadanému COM portu a ověří funkčnost tázáním se na napájecí napětí motoru X. Pokud výsledkem není nula, motor odpověděl a komunikace funguje. Následně se objeví okno s informací o úspěšném provedení připojení. V opačném případě je COM port uzavřen a je vypsáno hlášení o neúspěšném připojení.

6.2 Kalibrace

Pro možnost polohování je nejprve nutné kalibrací zařízení zjistit rozsah poloh a výchozí hodnoty pozic.

1. Motoru X je zapnuto monitorování signálu levého dorazového čidla a po sepnutí výkonového okruhu je proveden požadavek na neustálé otáčení se hřídele směrem doleva. Jakmile se změní logický stav dorazového čidla, motor se ihned zastaví. Během otáčení je vytvořena programová smyčka, při které se testuje aktuální rychlost. Pokud je nulová, motor byl zastaven dorazovým čidlem a nyní se nachází v levé krajní pozici, které je nově přiřazena nulová hodnota. Zároveň je vypnut výkonový stupeň a je vymazána chyba vzniklá aktivací dorazového čidla.

```
Call Leve_Cidlo
  MDC1.Setting.EnablePowerAmplifier 1
  MDC1.OperatingModes.ProfileVelocity 1, Write_Parameter, -60

Kalibrace_LIML:
  If Val(MDC1.ReadValues.Speed_ActValue(1)) = 0 Then
    MDC1.Setting.DisablePowerAmplifier 1
    MDC1.Setting.FaultReset 1
    MDC1.Config.CallByIndex 1, Write_Parameter, 40, 3, 0
  Else
    GoTo Kalibrace_LIML
  Exit Sub
  End If
```

2. Nyní je aktivováno pravé dorazové čidlo (tím je monitorování levého čidla deaktivováno) a obdobným způsobem je zjištěna pravá krajní pozice, ve které je motor zastaven.
   Do proměnné *Ref_poziceX* je uložena aktuální pozice motoru a přepočtem na milimetry (počet kroků / 250) je zjištěna velikost rámu v ose X se zaokrouhlením na jedno desetinné místo. Protože z neznámých příčin docházelo k chybnému čtení pozice motoru v oblastech kolem 200 mm, je nyní v pravé

krajní pozici opět nově přiřazena nulová hodnota. Tím je zajištěn pohyb motoru od záporné velikosti *Ref_poziceX* do nuly (obr. 6.2). V tomto rozmezí probíhá čtení pozic v pořádku.

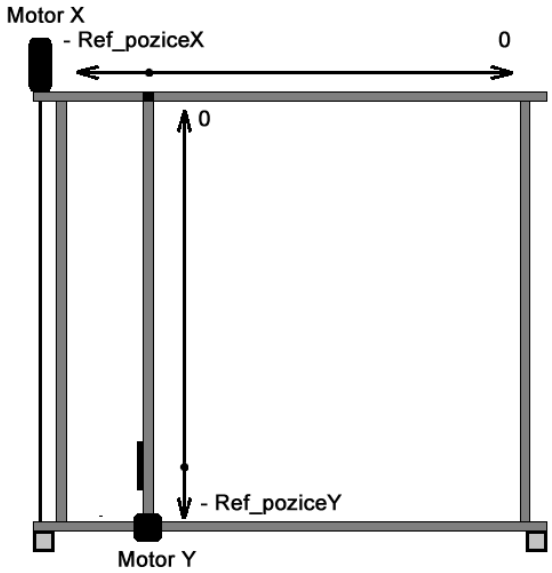
```
Label_VelikostX = Round(((MDC1.ReadValues.Position_ActValue(1))) / 250, 1)
Ref_poziceX = MDC1.ReadValues.Position_ActValue(1)
MDC1.Config.CallByIndex 1, Write_Parameter, 40, 3, 0
```

3. Obě čidla jsou vypnuty a motoru X je zadán pojezd do pozice -*Ref_poziceX*, čímž se zarovná do levé (výchozí) pozice. Opět použitím smyčky se zjišťuje aktuální rychlost a pokud je nulová, je vypnut výkonový okruh, odpojen motor X a připojen motor Y.

```
Call Vypni_Cidla
  MDC1.OperatingModes.ProfilePosRefValueSpeed 1, Write_Parameter, 60
  MDC1.Setting.EnablePowerAmplifier 1

Zarovnani_doleva:
  MDC1.OperatingModes.ProfilePosAbsolute 1, Write_Parameter, (-Ref_poziceX)
    If Val(MDC1.ReadValues.Speed_ActValue(1)) = 0 Then
      MDC1.Setting.DisablePowerAmplifier 1
      MDC1.Setting.FaultReset 1
    Else
      GoTo Zarovnani_doleva
    Exit Sub
    End If
```

4. Kalibrování v ose Y probíhá shodně jako v případě kalibrování osy X. Pro uchování výchozí (dolní) pozice je zvolena proměnná *Ref_poziceY*.



Obr. 6.2: Nákres rámu polohovacího zařízení a vyznačení pozic.

5. Dokončení kalibrace je zahlášeno dialogovým oknem, po jehož zavření se aktivují tlačítka pro možnosti polohování a vypíše se zjištěná velikost rámu.

6.3 Poloha osy X a osy Y

Samotné ovládání motoru X je možné provést tlačítkem „Poloha X“. Monitorování dorazových čidel je kvůli zjednodušení programu nyní deaktivováno. V aplikaci je tento stav ošetřen takovým způsobem, aby nebylo možné zadat pozici větší, než je velikost rámu. Zároveň je zjišťováno, pokud byla zadána pouze čísla. O neplatně zadané pozici nebo rychlosti je uživatel informován vyskakovacím oknem. Po stisku tlačítka je provedeno připojení k motoru X, zapnutí jeho výkonového okruhu, dále jsou čteny zadané hodnoty požadované rychlosti posuvu [mm/s] a pozice [mm], jsou převedeny na počet otáček za minutu, resp. počet kroků. Poměr otáčení [ot/min] vzhledem k rychlosti posuvu [mm/s] je 0,75. Od převedené pozice (zadaná pozice * 250) je třeba vždy odečíst hodnotu *Ref_poziceX*. Polohování motoru Y se děje obdobným způsobem.

6.4 2D Polohování

Při zadávání různých po sobě jdoucích pozic by měla být dodržena konstantní výsledná rychlost posuvu. Proto je nutné vypočítat dílčí hodnoty rychlostí posuvů v ose X a Y.

Princip výpočtu je velmi jednoduchý. Koncový bod je zadán uživatelem do textových polí „Text_PolohaX“ a „Text_PolohaY“, rychlost je zadána do pole „Text_RychlostXY“. Výchozí bod je určen souřadnicemi uloženými v proměnných *VychoziX* a *VychoziY*, jejichž hodnoty jsou získány čtením aktuální pozice motoru X, respektive motoru Y. Zároveň se čtením aktuální pozice je sepnut výkonnový okruh motoru, aby byl motor připraven pro následovné polohování.

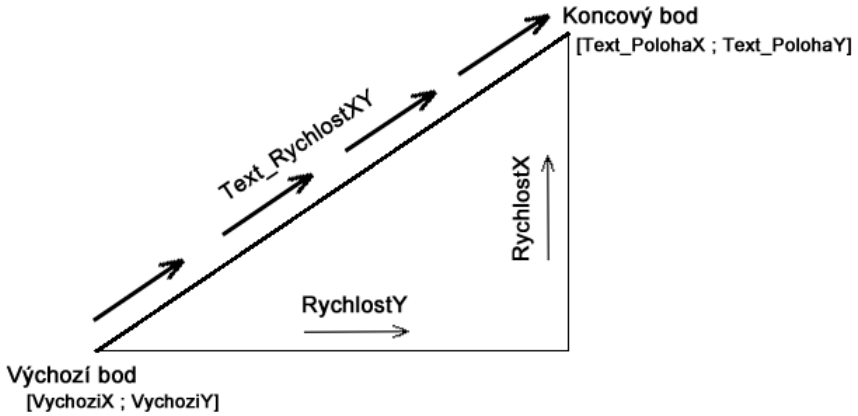
```
Call PripojX
  Call Vypni_Cidla
  VychoziX = (((MDC1.ReadValues.Position_ActValue(1)) + Abs(Ref_poziceX)) / 250)
  MDC1.Setting.EnablePowerAmplifier 1

Call PripojY
  Call Vypni_Cidla
  VychoziY = (((MDC1.ReadValues.Position_ActValue(1)) + Abs(Ref_poziceY)) / 250)
  MDC1.Setting.EnablePowerAmplifier 1

Call Vypocet
```

Vzniklý pravoúhlý trojúhelník je naznačený na obrázku č. 6.3. Velikosti přepony trojúhelníku je přidělena zadaná rychlost společného posuvu („Text_RychlostXY“). Tato rychlost se rozdělí v poměru jednotlivých odvěsen vůči přeponě. Pro výpočet délky přepony je využito Pythagorovy věty, velikost odvěsny je vždy rovna rozdílu dílčích souřadnic koncového a výchozího bodu. Před samotným výpočtem je provedena kontrola platně zadaných hodnot. Celý výpočet je uveden v následujícím zdrojovém kódu. Pokud je velikost přepony nulová, nebyly zadány nové souřadnice a není provedena žádná akce.

```
Prepona = (((Val(Text_PolohaX)-VychoziX)^2+(Val(Text_PolohaY)-VychoziY)^2)^0.5)
  If (Val(Prepona) <> 0) Then
    RychlostX = (((Val(Text_PolohaX)-VychoziX)/Prepona)*Val(Text_RychlostXY))
    RychlostY = (((Val(Text_PolohaY)-VychoziY)/Prepona)*Val(Text_RychlostXY))
  Else
  Exit Sub
  End If
```



Obr. 6.3: Vyznačení proměnných pro výpočet rychlostí posuvů.

Zjištěné hodnoty rychlosti jsou společně se zadanými souřadnicemi koncového bodu použity pro příkaz k posunu. Výpočet pracuje s hodnotami v milimetrech, hodnota pozice je proto opět převedena na počet kroků. Rychlost posuvu musí být zadána v absolutní míře, motor pak volí směr otáčení dle zadané pozice. Nejprve je připojen motor Y, kterému je nejprve zadána příslušným parametrem rychlost polohování a pak teprve pozice. Motor Y je nyní v pohybu. Program se ihned připojí k motoru X a stejným způsobem zadá hodnotu rychlosti a pozice.

```
Call PripojY
  MDC1.OperatingModes.ProfilePosRefValueSpeed 1,_
    Write_Parameter, (Abs(RychlostY) * 3 / 4)
```

```
  MDC1.OperatingModes.ProfilePosAbsolute 1,_
    Write_Parameter, ((Text_PolohaY * 250) - Ref_poziceY)

Call PripojX
  MDC1.OperatingModes.ProfilePosRefValueSpeed 1,_
    Write_Parameter, (Abs(RychlostX) * 3 / 4)
  MDC1.OperatingModes.ProfilePosAbsolute 1,_
    Write_Parameter, ((Text_PolohaX * 250) - Ref_poziceX)
```

6.4.1 Seznam pozic

Okno „2D Polohování“ obsahuje možnost zadat více pozic, které budou postupně po sobě automaticky nebo manuálně přidělovány motorům. K zadání pozic slouží textové pole s názvem „Text_Data“. Pozice se zadávají ve tvaru „$X;Y\ X;Y$“.

1. Zadaný řetězec souřadnic je nejprve nutné dekódovat. Počet zadaných pozic je zjištěn z počtu vyskytujících se znaků „;“ v poli „Text_Data“. Funkce „Trim“ ořízne zadaný řetězec v případě výskytu prázdných znaků na začátku nebo na konci řetězce a funkcí „Split“ je do proměnné *tmp* uloženo pole oddělených pozic. Funkce „UBound“ navrátí z proměnné *tmp* nejvyšší hodnotu pole, která odpovídá počtu pozic a je uložena do proměnné *Pocet_Pozic*. Informace o počtu pozic je zobrazena vlevo od textového pole pro zadávání pozic.

   ```
   tmp = Split(Trim(Text_Data), ";")
   Pocet_Pozic = UBound(tmp)
   Label_Pocet_Pozic = (Pocet_Pozic)
   ```

2. Následuje programová smyčka inkrementující proměnnou i až do počtu pozic. Výsledkem volání deklarované funkce „DekodujData“ jsou nalezené souřadnice v závislosti na zadání jejich pořadového čísla v poli „Text_Data“, který obsahuje pro oddělení prázdný znak. Nalezené souřadnice jsou uloženy do proměnné *PoleText* a zároveň do proměnné *Pozice* a jsou stále ještě odděleny znakem „;“. Pro samotné oddělení souřadnic na pozici X a pozici Y se opět využije funkce „DekodujData“, ale s použitím oddělovacího znaku „;“ vyhledávaném v proměnné *Pozice*. Výsledek je uložen přepsáním do proměnné *PoleText*, kde první pole obsahuje hodnotu odpovídající pozici X a druhé pole hodnotu pozice Y.
3. Proběhne inkrementace proměnné i a shodným způsobem jako při jednoduchém 2D Polohování popsaném výše je proveden výpočet rychlostí pohybu osy X a Y a následný pohyb. Do proměnných *VychoziX* a *VychoziY* se uloží aktuálně zadané souřadnice, které tak slouží k výpočtu při zadávání následujících souřadnic.

```
Smycka1:
If (i <= (Pocet_Pozic)) Then
  PoleText = DekodujData(i, Text_Data, " ")
  Pozice = PoleText
  PoleText = DekodujData(1, Pozice, ";")
  X = Val(PoleText)
  PoleText = DekodujData(2, Pozice, ";")
  Y = Val(PoleText)
  i = i + 1

Call Vypocet
Call Linearni_Polohovani
  VychoziX = X
  VychoziY = Y
...
```

4. Druhá smyčka neustále kontroluje aktuální rychlost pohybu v ose X, dokud není nulová. Následuje třetí smyčka, tentokrát zjišťující rychlost posuvu v ose Y. Jakmile jsou obě rychlosti nulové, byla dosažena zadaná pozice a program se vrací do první smyčky, která zajišťuje posun na další souřadnici. Pokud byla před kliknutím na tlačítko „Start“ zvolena možnost „Mauálně“, bude vždy zobrazeno vyskakující okno s informacemi o aktuální pozici a s výzvou k potvrzení pokračování na další pozici.

```
Smycka2:
Call PripojX
  If Val(MDC1.ReadValues.Speed_ActValue(1)) <> 0 Then
  GoTo Smycka2
  Else
  Call PripojY

Smycka3:
  If Val(MDC1.ReadValues.Speed_ActValue(1)) <> 0 Then
  GoTo Smycka3
  Else
    If Check_Manual.Value = 1 Then
     M = MsgBox("Dosazena pozice " & i - 1 & "/" & Pocet_Pozic & "_
     [ " & X & " ; " & Y & " ] " & vbNewLine & vbNewLine & _
     "Pokracovat na dalsi?", vbYesNo + vbInformation, "Polohovani")
      If M = vbYes Then
      GoTo Smycka1
...
```

5. Za poslední dosaženou pozicí je vypnut výkonový okruh obou motorů a je opět zobrazeno vyskakovací okno informující o dokončení polohování.

6.4.2 Import souboru

Do textového pole „Text_Data“ je možné seznam pozic naimportovat z textového souboru, což odstraňuje nutnost opakovaného vypisování již vytvořených seznamů s pozicemi. Výběr souboru je realizován použitím komponenty „CommonDialog“. Po kliknutí na tlačítko „Import“ se otevře klasické dialogové okno pro zvolení souboru. Filtrem je zařízen možný výběr pouze textových souborů. Do textového pole „Text_Data“ je obsah souboru přenesen postupně použitím smyčky. Dokud není dosažen konec souboru, do proměnné *hld* je ukládán postupně každý symbol z textovém souboru, který je jako řetězec spojován dohromady se symbolem následujícím. Do pole „Text_Data“ je přepsán obsah proměnné *hld* bez dvou posledních znaků, které přibyly postupným nabýváním proměnné *hld*, ke které byl vždy přiřazen následujícící znak a znak pro nový řádek v případě přetečení.

```
Do Until EOF(1)
  Line Input #1, chk
  hld = hld & chk & vbNewLine
Loop

Close #1
hld = Left(hld, Len(hld) - 2)
...
```

6.4.3 Časovač

V pozadí aplikace neustálě běží časovač, který každou vteřinu zjišťuje a vypisuje aktuální pozici motoru X i motoru Y. Byla tak vytvořena zpětná vazba při jednoduchém polohování na jednu pozici. Během automatického polohování je bohužel program zaneprázdněn a aktuální čtení pozic nefunguje.

```
Call PripojX
  Label_Aktualni_PoziceX = Round(((MDC1.ReadValues.Position_ActValue(1))_
  + Abs(Ref_poziceX)) / 250, 1)

Call PripojY
  Label_Aktualni_PoziceY = Round(((MDC1.ReadValues.Position_ActValue(1))_
  + Abs(Ref_poziceY)) / 250, 1)
```

7 ZÁVĚR

Pro zprovoznění dálkového ovládání lineární os Berger Lahr PAS41BR pomocí osobního počítače bylo zapotřebí vyrobit převodník umožňující komunikaci prostřednictvím USB portu počítače a RS-485 standardu na straně krokového motoru. Převodník využívá integrovaných obvodů FT232RL a MAX481CSA, pro jeho návrh byl použit program EAGLE ver.5.2. Osazený plošný spoj převodníku byl vestavěn do kompaktní plastové krabičky. Dalším z hlavních cílů této práce bylo porozumění základním příkazům prováděných motorem a proniknutí do jejich datové struktury.

Instalací patřičných ovladačů a použitím knihovny MDC_ActiveX byla vytvořena jednoduchá řídicí aplikace pro mapování akustického tlaku, která umožňuje po kalibraci zařízení současně i zvlášť ovládat krokové motory. Protože se jedná o sériovou komunikaci, není možné oba motory ovládat naprosto současně, což se projevuje mírným zpožděním pohybu v ose X oproti ose Y.

Důležitou vlastností aplikace je výpočet dílčích rychlostí při současném pohybu lineárních os, čímž je zajištěna konstantní rychlost pohybu měřicí sondy. Hlavní funkcí aplikace je možnost zadání seznamu pozic, které mohou být postupně dosaženy zařízením automaticky (vhodné pro měření akustického výkonu metodou skenování), nebo mohou být dosaženy poloautomaticky vždy s výzvou k dosažení následující pozice (vhodné pro metodu měření akustického výkonu v bodech).

Během provádění automatického polohování je, bohužel kvůli nutnosti neustálé kontroly rychlosti motorů, řídicí program zaneprázdněný a do ukončení polohování neposkytuje žádnou zpětnou vazbu. Toto chování je potlačeno možností poloautomatického polohování. Knihovna MDC_ActiveX a některé další použité knihovny a ovladače jsou kompatibilní pouze pro 32-bit operační systémy, čímž je znemožněna celková funkčnost řídicí aplikace pod modernějšími 64-bit operačními systémy.

LITERATURA

[1] *Product manual : Lexium Integrated Drive* [online]. Germany: Schneider-electric, 2008 [cit. 25. 5. 2013]. Dostupné z URL: <http://www.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/29000/FA29355/es_ES/ILS1B_ILS1F_ILS1R_manual_V201_EN.pdf>.

[2] *ILx1R RS485 Fieldbus Interface user Manual* [online]. Germany: Schneider-electric, 2008 [cit. 25. 5. 2013]. Dostupné z URL: <http://www.global-download.schneider-electric.com/852577A4005D7372/all/444D3BDC05CCCE00852577ED006862CE/$File/ilx1r_rs485_manual_v201_en.pdf>

[3] *MAX481/MAX483/MAX485/MAX487-MAX491/MAX1487*: Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers [online]. USA: Maxim Integrated Products, 2009 [cit. 25. 5. 2013]. Dostupné z URL: <http://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf>

[4] *FT232R USB UART IC* [online]. UK: Future Technology Devices International Ltd., 2010 [cit. 25. 5. 2013]. Dostupné z URL: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf>

[5] Bürkle, P. *Manual MDC_ActiveX*: MultiDeviceControl for Berger Lahr devices [online]. Berger Lahr, 2007 [cit. 25. 5. 2013]. Dostupné z URL: <http://www.4shared.com/office/cozjcuIm/MDC_ActiveX_BL_Manual.html>

[6] Smetana, C. a kol. *Hluk a vibrace, měření a hodnocení.*. Sdělovací technika, Praha: 1998. ISBN 80-901936-2-5.

[7] Brothánek, M. a kol. *Měření akustického výkonu pomocí akustické intensity* [online]. [cit. 25. 5. 2013] Dostupné z URL: <http://archiv.otevrena-veda.cz/users/Image/default/C1Kurzy/Fyzika/22_brothanekc.pdf>

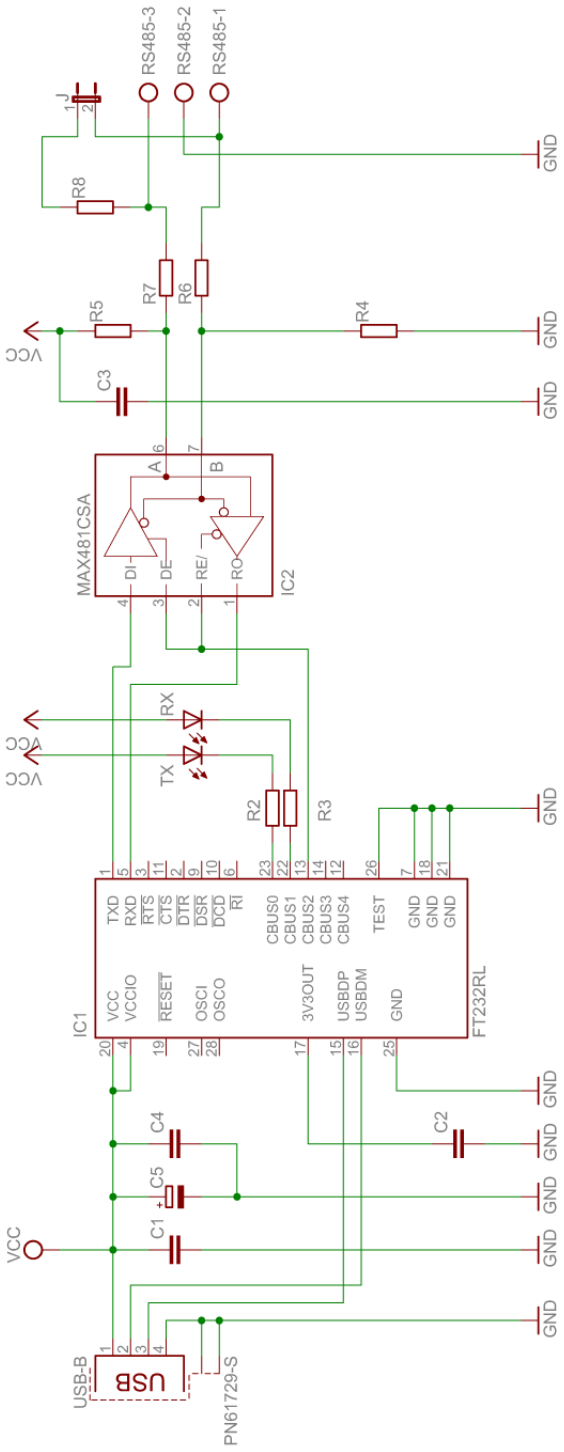
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A	Ampér, jednotka elektrického proudu
ASCII	americký standard kódování
b	bit, dvojková číslice
Bd	Baud, jednotka modulační rychlosti
DIP	druh pouzdra elektronických součástek
EEPROM	elektricky mazatelná paměť
I/O	vstupní / výstupní brána
LED	světlo emitující dioda
OFF	symbol pro stav vypnutí
ON	symbol pro stav zapnutí
PLC	programovatelný logický kontrolér
RS-485	standard sériové komunikace
SMD	elektronická součástka pro povrchovou montáž
SO	druh pouzdra integrovaných obvodů
SSOP	druh pouzdra integrovaných obvodů
UART	typ sériové komunikace, synchronní nebo asynchronní
USB	univerzální sériová sběrnice
V	Volt, jednotka elektrického napětí

SEZNAM PŘÍLOH

A KONSTRUKCE PŘEVODNÍKU

A.1 Schéma zapojení



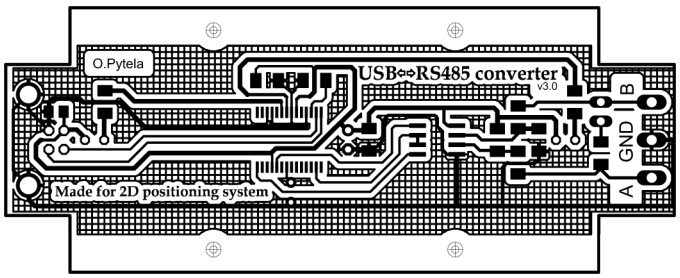
Obr. A.1: Schéma zapojení převodníku.

A.2 Seznam součástek

Tab. A.1: Seznam součástek.

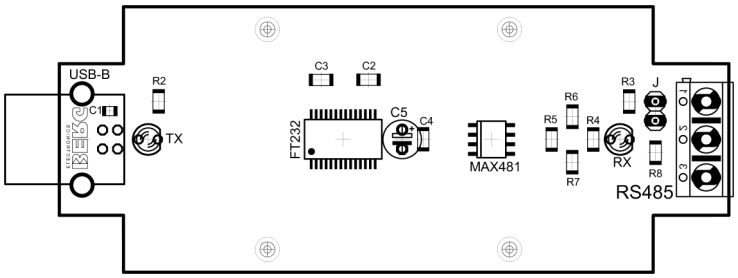
Název	**Hodnota**	**Pouzdro**	**Popis**
R2,R3,R4,R5	560Ω	SMD1206	Rezistor 1%
R6,R7	47Ω	SMD1206	Rezistor 1%
R8	120Ω	SMD1206	Rezistor 1%
C1			Neosazen
C2,C3,C4	100nF	SMD1206	Keramický
C5	4,7μF	5x7/2,5	Elekt. 50V
TX	LED	5mm	Žlutá
RX	LED	5mm	Červená
J	Jumper	2,5mm	
USB-B	USB-B	PN61729	USB-B konektor
RS485	Svorkovnice	AK500/3	Šroubovací
IC1	FT232RL	SSOP28	Převodník
IC2	MAX481CSA	SO8	Převodník

A.3 Předloha pro výrobu plošného spoje



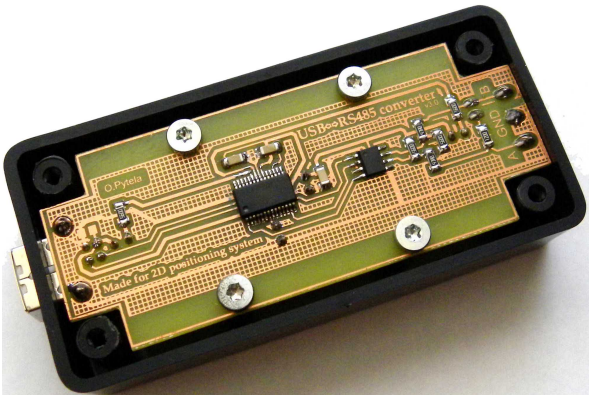
Obr. A.2: Předloha pro výrobu plošného spoje.

A.4 Osazovací plán plošného spoje



Obr. A.3: Plán rozmístění součástek.

A.5 Vyrobený převodník



Obr. A.4: Hotový odkrytovaný převodník.

B ZACHYCENÍ KOMUNIKACE

```
<20121129121215.031 TX>
###########01 [len=13]    Adresování motoru, adresa 01
<20121129121215.109 RX>
#01 [len=3]               Odpověď #01 >> Navázána komunikace
<20121129121221.156 TX>
0206001C00000000 [len=16] 28:6 Motion.invertDir-směr otáčení ?
<20121129121221.171 RX>
0A06001C00000001 [len=16] Výchozí směr proti směru hod. ručiček
<20121129121221.296 TX>
021A001D00000000 [len=16] 29:26 Motion.acc-nastavena akcelerace?
<20121129121221.328 RX>
0A1A001D00000256 [len=16] 256h=598 >> 598inc/min
<20121129121221.828 TX>
0205002300000000 [len=16] 35:5 PTP.v_tarPTP-referenční rychlost?
<20121129121221.843 RX>
0A05002300000064 [len=16] 64h=100 >>100ot/min
<20121129121248.671 TX>
0901001C00000002 [len=16] 28:1 Commands.driveCtrl -
<20121129121248.703 RX>        zapnutí výkonového okruhu
0101001C00000002 [len=16] Výkonový okruh připojen
<20121129121248.781 TX>
0207002000000000 [len=16] 32:7 Status.StopFault - výpis chyb
<20121129121248.796 RX>
0A07002000000000 [len=16] žádné chyby v záznamu
<20121129121248.875 TX>
0214001F00000000 [len=16] 31:20 Status.UDC_act - napájecí napětí ?
<20121129121248.906 RX>
0A14001F0000012C [len=16] 12Ch=300 >> 30,0V
<20121129121248.921 TX>
0219001F00000000 [len=16] 31:25 Status.TPA_act - teplota motoru ?
<20121129121248.953 RX>
0A19001F0000002F [len=16] 2Fh=47 >> 47°C
<20121129121254.703 TX>
090500230000012C [len=16] 35:5 PTP.v_tarPTP - rychlost nastav
<20121129121254.718 RX>        (12Ch=300) >> 300ot/min
010500230000012C [len=16] Provedeno
<20121129121221.625 TX>
0201002300000000 [len=16] 35:1 PTP.p_absPTP - absolutní poloha ?
<20121129121221.656 RX>
```

```
0A01002300004E20 [len=16] 4E20=20000h >> 20000 kroků
<20121129121245.484 TX>
09030023000003E8 [len=16] 35:3 PTP.p_relPTP - relativní poloha
<20121129121245.515 RX>        (3E8h=1000) + 1000inc
0703002300000137 [len=16] Provádím příkaz, aktuální poloha
<20121129121253.515 TX>   (137h=311) + 311inc
09030023000003E8 [len=16] 35:3 PTP.p_relPTP - další posunutí
<20121129121253.546 RX>   o 1000inc
01030023000003E8 [len=16] Příkaz rel. polohování proveden
<20121129121252.765 TX>
0206001F00000000 [len=16] 31:6 Status.p_act - aktuální poz. motoru?
<20121129121252.781 RX>
0A06001F000055F0 [len=16] 55F0h=22000 > 22000Inc
```

C OBSAH CD

- V kořenovém adresáři je uložena tato práce „**2D_Polohovaci_Zarizeni.pdf**“.
- Adresář „**Program**“ obsahuje:
  - zdrojové soubory aplikace k řízení polohovacího zařízení (Visual Basic 6),
  - ve složce „**EXE**“ je vytvořená aplikace „**2D_Polohovaci_Zarizeni**“,
  - složka „**Import**“ obsahuje textový soubor „**Priklad_importu**“ s příkladem dat k importování pozic.
- Adresář „**Ovladače**“ obsahuje:
  - knihovny potřebné pro spuštění aplikace a návod k instalaci „**Navod.txt**“.
  - Složka „**Setup**“ obsahuje soubory pro spuštění instalace,
  - adresář „**FTDI**“ obsahuje soubory potřebné pro instalaci ovladačů k převodníku.
- Složka „**Test**“ obsahuje textový soubor „**Pohyb_motorem**“ pro možnost ovládání motorů např. pomocí Hyperterminálu.